IRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P 105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Rapports de Recherche

N° 127

A NEW DATABASE QUERY LANGUAGE FOR NON-PROFESSIONAL USERS: DESIGN PRINCIPLES AND ERGONOMIC EVALUATION

Alain MICHARD

Avril 1982

A NEW DATABASE QUERY LANGUAGE FOR NON-PROFESSIONAL USERS: DESIGN PRINCIPLES AND ERGONOMIC EVALUATION.

A. MICHARD

I.N.R.I.A.
Sophia Antipolis
06560 - VALBONNE - FRANCE.

ABSTRACT

A new query language designed to improve ease-of-use and ease-of-learn for "naive" users is presented. Its main interest is to avoid the explicit use of boolean operators for set operations by pointing on Venn diagrams. A human factors comparison with a more traditional design has been performed, using a query writing task. Results show that graphic representation of selected subsets allows less error prone queries in a single relation database.

RESUME

Un nouveau langage d'interrogation de base de données concu pour améliorer la facilité d'apprentissage et d'utilisation par des utilisateurs occasionnels est présenté. Son principal intérêt est d'éviter l'usage explicite des opérateurs booléens lors des manipulations ensemblistes, grâce à des désignations sur des diagrammes de Venn. Une évaluation ergonomique du nouveau language montre que la représentation graphique des sous-ensembles sélectionnés permet une diminution sensible du nombre de requêtes erronées. Les résultats sont limités à l'interrogation d'une base de données mono-tabulaire.

1. INTRODUCTION.

1.1. Why a new query language?

With the development of office automation, an increasing number of occasional or "naive" users try to use database systems for their personal or institutional documentation needs. For the system designer, the main characteristic of this population is that they would not spend much time learning a tortuous command language, and that even if they did they would soon forget it having but few occasions on which to consolidate their knowledge.

For these users none of the existing query languages is really satisfactory. One of the most well known named "Query-by-Example" (QBE), was indeed designed to improve ease-of-use and ease-of-learn for non-professional users (Zloof,1975 ; Thomas and Gould,1974).In spite of this preoccupation Thomas and Gould in their ergonomic evaluation of QBE had to provide their subjects with an intensive training period of 4 hours and 35 minutes including two 20-question test phases. It worth noting that despite this training period one third of the subjects' queries expressed in QBE in the final test were erroneous. Furthermore, after a two-week delay the proportion of correct queries dropped down to 53 percent (mean score for six subjects).

For other query languages, the situation seems even worse: for example in her study of the SQUARE and SEQUEL languages, Reisner (1975,1981) had to teach non-programmer students for 14 hours to obtain a mean percentage of "essentially" correct answers on final exam of 65 percent for SEQUEL and 55 percent for SQUARE. Of course comparisons of

these different results should be made very cautiously,as stated by Reisner: tasks used to measure ease-of-use and methods to teach the subjects differ among the different studies.

1.2. Design principles.

There are two tendencies in design efforts toward improved query languages. The first deals with the interpretation of queries expressed in natural or nearly natural language. Some of the research topics are:

- query analysis by a semantic grammar to translate a "nearly natural" language query into a strict boolean mapping function which can be interpreted by a classical DBMS. (Hendrix, Sacerdoti, Sagalowicz and Slocum, 1978).
- interpretation of fuzzy queries of the kind "age=old" or "price=cheap" (Tahani,1977)
- use of information structures derived from Quillian's model of human semantic memory. The goal is to create "intelligent" databases able to dialogue with a user about a given topic (Schank Kolodner and DeJong, 1980).

Thomas (1976) and Shneiderman (1978) give strong arguments to support the fact that natural language does not generally lead to precise and unambiguous queries. It seems that these systems might be useful to give the users the possibility to browse through the database without a precise knowledge of its structure. But if the task requires an exhaustive and accurate selection, boolean manipulations are needed.

The other research tendency is to make hypotheses on the origin of the difficulties encountered by beginners to learn and use existing

query languages. To do this we can make use of the previous ergonomic studies on query languages (Reisner 1981, is a good "state of the art" for this domain), or transpose results of the general man-machine communication research field (good introductions should be here (Martin 1973) and (Shneiderman 1980)). We think that some of the most important and well established principles for query languages are:

-non-procedurality; means that the user has to indicate what the selection he want to obtain is, but not the steps to reach this result. Thomas and Gould give strong arguments in favor of this design principle.

-immediate feed-back to every keystrike; the system must provide an immediate answer showing the potential effect of any command when issued.

-free-order for specifying the selection criteria; this is one of the "good features" of QBE, and we think it worth keeping.

-aid in the manipulation of boolean operators for set operations. The difficulty for "naive" users to manage with some boolean operators -disjunctive AND, NOT- is well known. Our query language will provide very great help for these operations using Venn diagrams to materialize selected subsets. Thomas 1976, has shown that the use of such diagrams might reduce errors in set manipulations.

-menu-oriented dialogue; the menu technique has the considerable advantage of reducing or avoiding clerical errors (Martin,1973).

-external representation of the data structure; this is another good point for QBE that we think must be conserved.

2. DESCRIPTION OF THE "GRAPHICAL QUERY LANGUAGE" (GQL).

We are interested by database systems embedded in Text Management systems. Users of such systems make frequent use of very simple databases. Their typical need is to select items in customers, suppliers or subscribers files in order to prepare mailing lists. We shall make the hypothesis that to meet the great majority of demands to these small office systems, it is sufficient to permit manipulations of databases that can be externally represented to the users as a single relation (one table).This means in fact that we shall not authorize links between relations in a mapping function.

The easiest way to present GQL is probably to do so using an example. Let us consider a database containing information on subscribers of a review. It can be presented as a table, each line containing Name, Address, Region- number (RNBR) (*), First subscription date (FSD), Last subscription date (LSD), subscription duration (SDUR), and probably a few more in reality.

When someone want to query this database, (SELECT function of a general menu for instance) the menu presented at Figure 1 appears on his screen. Note that the three circles in the lower right corner are not present at this stage. The upper line of this menu is a representation of the data structure giving the variables names (columns names). Three important pointing areas are also displayed:

-relational operators area =,#,<,>,<=,>=

(*) French administrative regions ("départements") are numbered from 1 to 95.

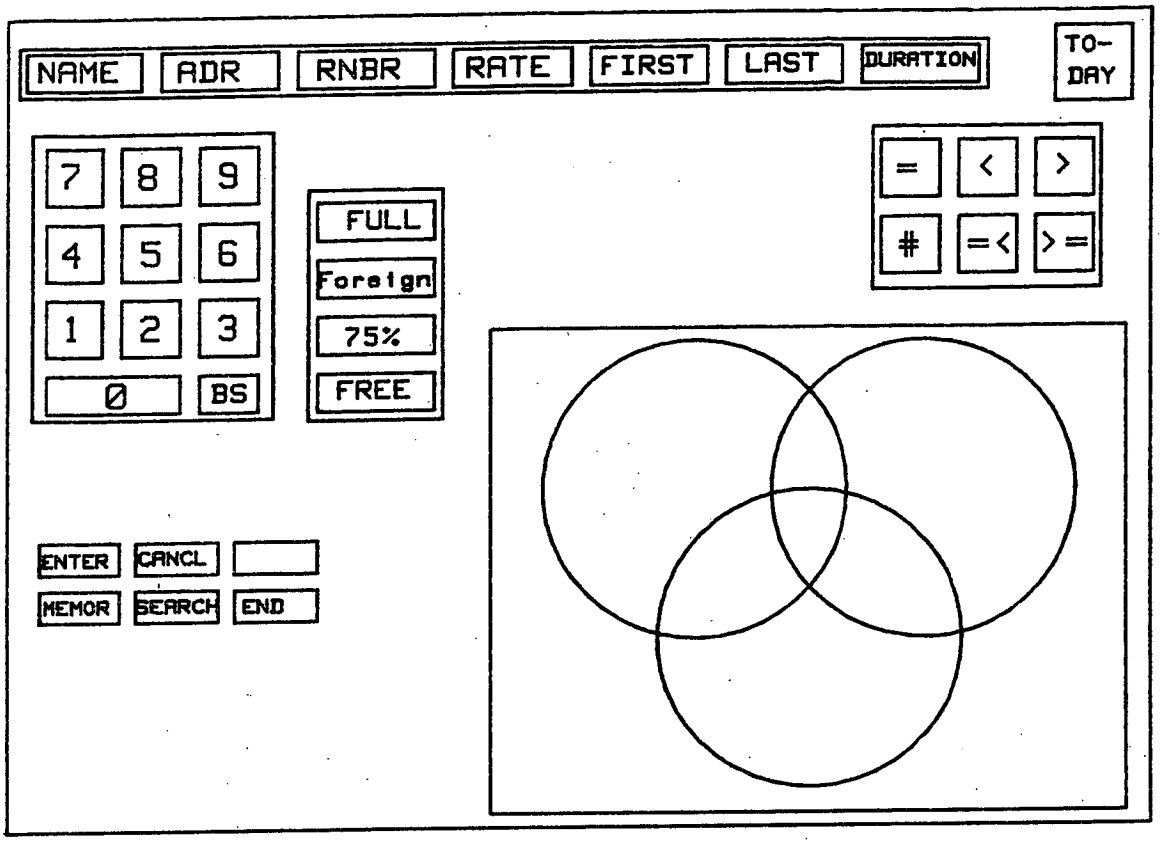

FIG.1: GQL menu for the "subscribers" database. The three circles are not present at the beginning of a query formulation.

-constants areas. The first one is a numeric pad . The second is a rate value area used to enter subscription rates without using a keyboard. "To-day" is a special soft-key providing the current date.

-commands area: Enter, Cancel, Memorize, Search, End.

One can note the absence of any boolean operators and parenthesis area.

Consider now the query :"Select everyone in the base living in region 75 or in regions 78 and over, and who is subscribing at full rate."

Figure 2 summarizes the steps the operator must go through to issue this query. The three selection criteria (RNBR=75, RNBR>=78, Rate=Full) might be entered in any order. Each time a criterion is entered a circle is drawn in the display area with a legend indicating the selection to which it corresponds.

When the three criteria have been entered, the operator designates which elementary subsets he wants to select by pointing (with any convenient pointing device)on the Venn diagram. Hatching provides him feedback on his choices. The two pointings necessary to specify the query given as an example might of course be done in any order.Afterwards, a pointing on "SEARCH" will validate the selection and launch the query process.

The number of simultaneous criteria that can be treated in an elementary query has been limited to three, for pointing on Venn diagram involving more than three circles would become quite involved. To deal

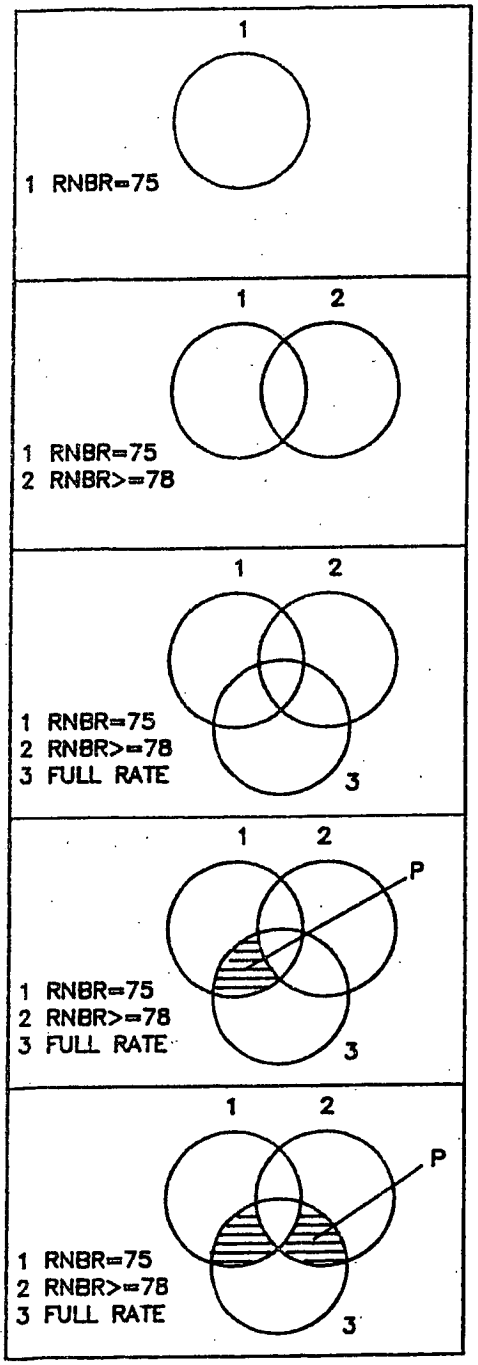

Figure 2: Display area at each step of a query formulation.

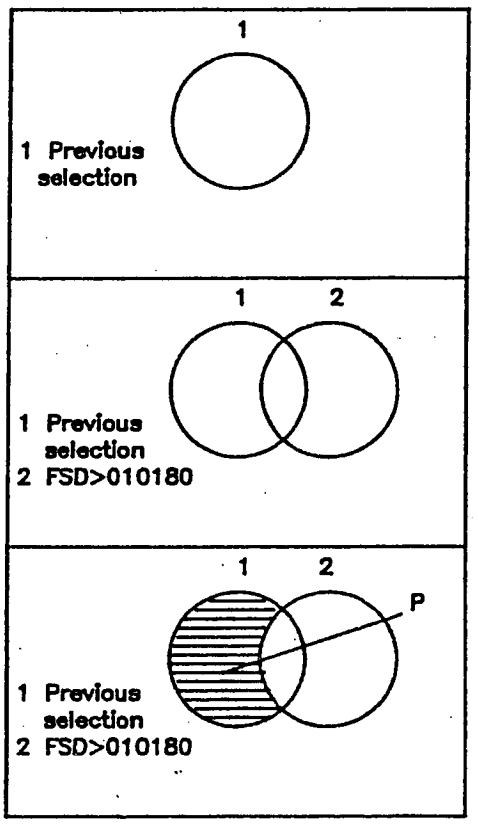

Figure 3: Processing a complex query.

with more complicated queries the operator must use the "MEMORize" function. For example let us consider the query "Select everyone living in regions 75, 78 and over, subscribing at full rate, and subscribing before 01.01.80. The first steps of this query process are the same as for the previous one. Instead of pointing on "SEARCH" the operator will point on "MEMOR" and obtain a first circle representing his first selection. Figure 3 illustrates the next steps: in this example the operator chooses to exclude every tuple corresponding to the criterion "FSD>010180". Instead of this he could have entered the criterion "FSD<010180" and pointed on the intersection. The process ends with pointing on the "SEARCH" soft-key as in the previous case.

The general rule for complex queries is that MEMOR key causes all previous selected subsets to be represented by a single circle in location 1. This new subset can be then combined with two new criteria. The function MEMOR stands for the traditional parenthesis in boolean equations, with the limitation that no more than three criteria can be present in the same parenthesis.

Two other language facilities are important, the first being the possibility to define a selection criterion by the comparison of variables: for example, the elementary condition "new subscribers" could be described as "FSD=LSD".

The second facility is the "Cancel" key which makes it possible at any moment of the dialogue to withdraw the effects of the last pointing (command, operator, constant, boolean choice).

3. ERGONOMIC EVALUATION OF GQL.

Absolute evaluation of a new query language being impossible for lack of a normative model of human query processing, a good way to have some idea of its practical interest is to compare subjects' performances using two different languages. Of course the problem is to decide by which features the two languages will differ from one another in order to usefully interpret the performance differences that will be eventually measured.

The main originality of GQL is perhaps the graphical aid provided to users to form boolean mapping functions. We have chosen to compare GQL to an ad hoc query language (the "TEST language") which differs only from the former in that the user has to describe set operations by classical explicit boolean operators (AND, OR, MINUS, and parenthesis). Figure 4 shows the menu proposed to the user processing a query.

The two languages GQL and TEST have been implemented on a graphic desktop computer, allowing queries on a small database (80 tuples). A graphic data tablet was used as pointing device.

3.1. Subjects.

The subjects were twelve French college students aged from 15 to 18, none of whom had computer knowledge. All French college students have in their curriculum the opportunity to learn basic boolean algebra and the use of Venn diagrams.

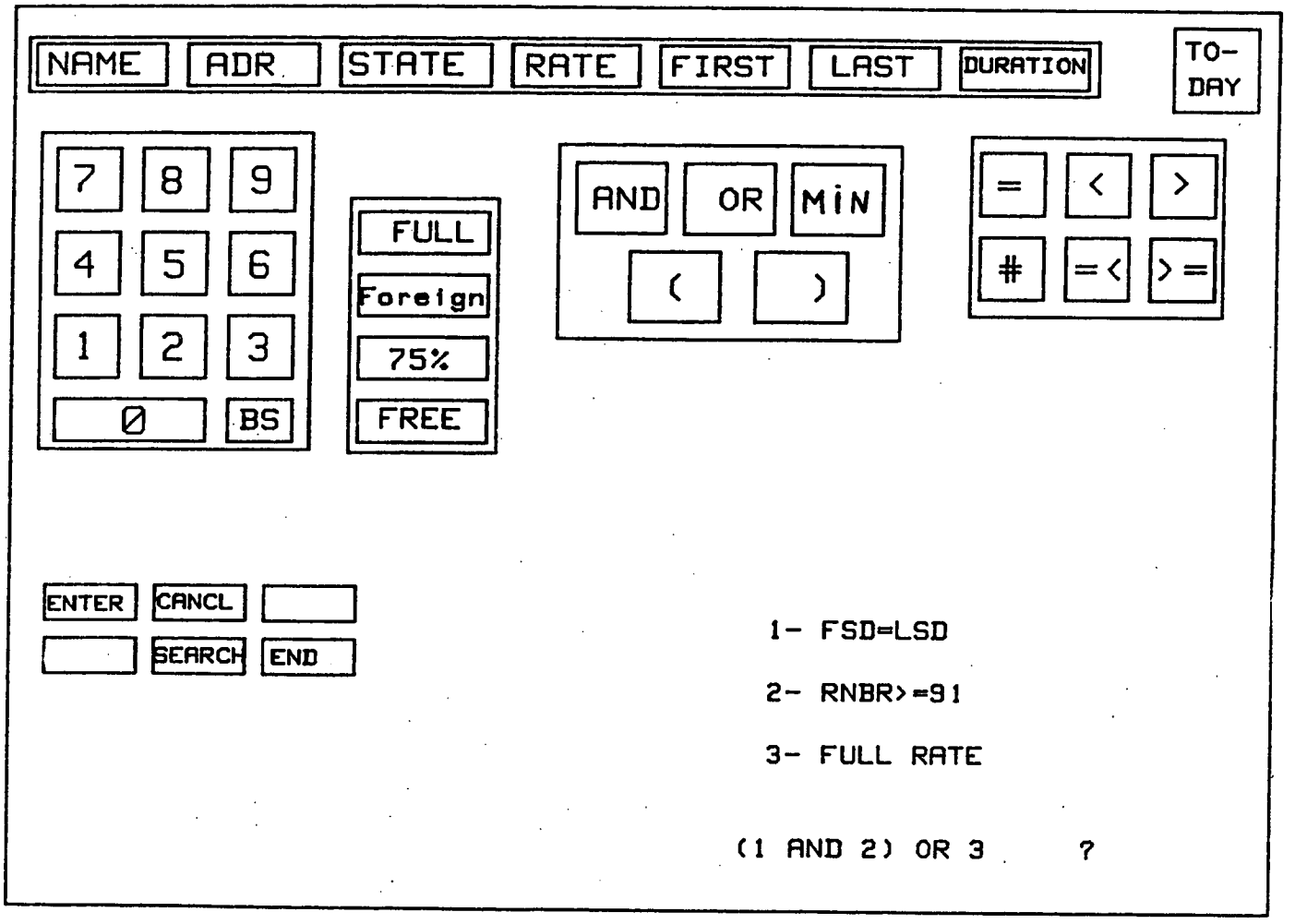

FIG. 4 : The TEST menu.

3.2. Training.

Every subject learned both languages. Learning was individual by mean of a 50 to 70 minute "hands-on-keys" session for the first language and a 30 to 45 minute session for the second one, a good part of the information to be learned -data organisation, task goal- being common to both situations.

The experimenter explained first the content of the database, its structure, and what is meant by a selection. Then he proposed 18 query problems in order of increasing complexity. During this query writing task, subjects were given all the help they needed each error being corrected and explained as it occured. First examples involved elementary relations between two subsets (AND, OR, MINUS). The following examples required a combination of three or four subsets.

Having learned a query language the subjects had to perform the experimental task for this language and then were trained in the other language.

To control learning transfer effects between the two languages the subjects were distributed among two groups, one group beginning with GQL and the other with TEST.

3.3. Experimental task.

For each language, each subject task was to translate 8 selection definitions given in natural language in a valid query. The 8 boolean equations corresponding to the 8 queries are given in Table 1. For each language the queries had the same boolean structure but a different content: for example, for a same structure "X AND Y" content 1 could be

Table 1: Boolean structure of query problems.

Problems	Structure
1 and 9	$A \cap B \cap C$
2 and 10	$A \cap (B \cup C)$
3 and 11	$A \cup (B \cap C)$
4 and 12	$(A \cup B) - C$
5 and 13	$(A - B) \cap C$
6 and 14	$A \cap B \cap C \cap D$
7 and 15	$(A \cup B) - (A \cap B)$
8 and 16	$(A \cap B) \cup (C \cap D)$

Table 2: Experimental design.

Languages	Problems 1 to 8	Problems 9 to 16
GQL	Group 1 (Subjects 1 to 6)	Group 2 (Subjects 7 to 12)
TEST	Group 2	Group 1

Table 3: Number of erroneous queries (boolean errors only) for the two languages and the eight problem structures. Maximum possible number in each square is 12.

	Problem structures								
	1	2	3	4	5	6	7	8	totals
GQL	0	2	0	1	2	1	1	3	10
TEST	0	3	4	4	8	1	9	7	36

Table 4: Taxonomy of set operation errors for each language.

	GQL	TEST
Missing pairs of brackets or MEMOR forgetting	6	19
Brackets in wrong place	-	9
Wrong operator	5	10
Missing criterion	2	6
Superfluous criterion	1	2
Others	2	3
Total	16	49

"RNBR=75 AND FULL-RATE" and content 2 "SDUR>12 AND FSD<12.31.78". For six subjects content one corresponds with GQL and content two with TEST, and vice versa for the other six subjects.

The problems were presented in a different random order for each subject and each language.

Table 2 illustrates the experimental design chosen to allow comparison among the two languages within the same group, and to test the effect of the type of boolean query structure on error frequency.

4. RESULTS.

4.1. What is scored as an "error"?

4.1.1. Multiple solutions problems.

For several problems there are several solutions leading to the right selection. For example, "Full_rate MINUS RNBR=75" is equivalent to "Full_rate AND RNBR#75". Every equivalent solution was considered correct as long as it was syntactically and semantically correct.

4.1.2. Bracketing.

Superfluous brackets in TEST queries or superfluous use of the MEMOR function in GQL were not considered as errors. On the other hand, missing brackets are generally erroneous exept if the general rules of boolean algebra authorizes such a simplification. There is not an explicit hierarchy between boolean operators. Parsing of boolean expressions in TEST program is made from left to right, but this fact is not known by the subjects and missing brackets are counted as errors even if the correct selection is made due to our implementation.

4.2. Errors in set operations.

Table 3 gives the number of erroneous queries for the two languages and the eight problem structures (maximum error number for each experimental situation is 12). TEST language gives rise to nearly four times as many errors than GQL, with strong differences among boolean structures. The difference between languages is statistically significant (F test after Fisher's angular transformation).

Three problems (1,2,6) are not prone to error for both languages, the five others being at the origin of the difference among languages. Table 4 gives a taxonomy of boolean algebraic errors. The total numbers of errors are greater than numbers of erroneous queries because in a number of cases there were several errors in the same query.

In the error analysis it is interesting to note that:

- AND operator seems to be the easiest one to use by the subjects: problems 1 and 6 involve only this operator and give rise to only very few errors.

- The origin of the difference of error score between the two languages can be found mainly in brackets misuses: in TEST expressions 19 pairs of brackets were missing or incomplete, and using GQL subjects forgot six times to use the MEMOR function and tried to enter simultaneously a fourth elementary criterion.

- An important difference was found between the languages with problem 7 which is an exclusive OR between two subsets. In GQL the designation is very simple (only one error) but the TEST expression (A U B) - (A $\cap$ B) is correctly given by only 3 subjects in 12.

4.3. Errors in the expression of elementary criteria.

There were very few errors in the expression of elementary criteria (5 over 300). It seems that the menu technique used in both languages is very efficient on this point.

4.4. Non-erroneous variations in producted queries.

The error analysis does not show an important difference between the two languages, that is the degree of variation between different correct queries. For a same problem correct TEST queries are identical for most subjects: it seems that they try to translate the query problem word after word from its expression in natural language to an equivalent in TEST. Elementary criteria appear in the same order in the boolean mapping function and in the natural language expression. However, variations were observed in query boolean structure for two problems: problem 4 was solved by a query of the kind "(A U B)-C" by three subjects, and by a query of structure "(A U B)∩D" by five others, as has been explained in 4.1.1. Problem 5 was solved with either the structure "(A - B) ∩ C" or "(A ∩ C) - B".

In GQL differences among query formulations for a same problem are very frequent. Subjects rapidly discovered that the elementary criteria definition order does not matter, and that the MEMOR function could be used on a two-criteria combination as well as on a three-criteria one. These language properties were widely used by subjects producing designation order variations.

5. CONCLUSIONS AND SUMMARY.

The results described above show that improvments are possible in the field of DBMS query languages with quite simpler methods than natural language understanding which will probably remain out of the reach of micro-computer based office systems during this decade.

High-school students without any computer knowledge performed a query writing task with a 90% success rate after a training period of about one hour. These performances were achieved by mean of a language design which we think derives its effectiveness from two main features:

- a tabular representation of the data structure, inspired from QBE, but that can be directly used as a variable selection menu.
- an interactive graphic assistance to subsets selection by mean of Venn diagrams. These diagrams are widely used in schools to learn boolean algebra. Our design shows that it can be a good support to boolean reasoning in other circumstances.

There are some obvious limitations to the presented GQL version. Some are specific to our implementation and could be easily eliminated in the futur. One would probably like to find a "COUNT" function giving the cardinal number of a selected subset, and arithmetic functions on numeric variables ("SUM", "MEAN"), and possibility to define more sophisticated combinations of variable values... In our database example it might have been useful to have the possibility of defining an elementary criterion of the kind: "Subscription_date + Subscription_duration < To_day + 30" which makes it possible to find subscriptions ending during the next 30 days. Such functionnalities would be very easy to provide.

Another limitation is inherent to our definition of GQL: the user can access only one relation at a time and cannot make use of links among relations in his queries even if these links have been defined in the database schema. We think that the available experience of QBE shows that this possibility when used by "naive" users is error prone and that most of the needs of small office systems users can be met using single relation data structures.

Nevertheless, despite these limitations GQL probably provides a noticeable improvment to database query language for casual users.

Acknowledments.

This work was done under a research contract from the Agence de l'Informatique, Projet Pilote Kayak.

References.

Hendrix,G.C., Sacerdoti,E.D., Sagalowicx,D., Slocum,J., 1978, Developing a natural language interface to complex data, ACM Transactions on Database Systems, 3, 105,147

Martin,J., 1973, Design of Man-Computer dialogues, Prentice Hall, Englewoods Cliffs, New Jersey

Reisner,P., Boyce,R.F., Chamberlin,D.D., 1975, Human factors evaluation of two database query languages- Square and Sequel, in Proceedings of the National Computer Conference, AFIPS Press, Arlington, Va.

Reisner,P., 1981, Human Factors Studies of Database Query Languages: A Survey and Assessment, Computing Surveys, 13, 1, 13-31

Schank,R.C.,Kolodner,J.L., DeJong,G., 1980, Conceptual Information Retrieval, Information Retrieval Research, Cambridge, UK. 94-116

Shneiderman,B., 1978, Improving the human factors aspects of database interactions, ACM Transactions on Database Systems, 3, 4, 417,439

Shneiderman,B., 1980, Software Psychology, Winthrops Publishers Inc., Cambridge, Mass.

Tahani,V., 1977, A conceptual framework for fuzzy query processing- A step toward very intelligent database systems, Information Processing & Management, 13, 289-303

Thomas,J.C., and Gould,J.D., 1974, A psychological study of query by example, IBM Research Report, RC5124.

Thomas,J.C., 1976, Quantifiers and question-asking, IBM Research Report, RC5886.

Thomas,J.C., 1977, Psychological issues in data base management, International Conference on Very Large Databases, Tokyo, 169-185

Zloof,M.M., 1975, Query by Example, Proceedings of the National Computer Conference, AFIPS Press, Arlington